Lecture 5 - Entity-Component-System

"Either make it so simple that there are obviously no deficiencies or make it so complicated that there are no obvious deficiencies." - (Tony Hoare, adapted)

- How to represent game objects and store them
- Logic behind each game object
- Managing a large number of game objects

Last Week Recap

- Event systems (Kahoot)

Game Objects and World

To start us off:

- What is a game object?
- What is a game world?
- Why do we need these concepts?

ECS (Entity-Component-System)

Architectural pattern used in many engines and games. Examples:

- Minecraft entt
- Overwatch ECS
- C++ 11 ECS EntityX
- Unity ECS

ECS - Entity

- **Everything** is an entity
- Without extra information an entity is like an object without fields and methods
- Essentially, a bag of components

ECS - Component

- Traditionally, holds only data (variations exist that also hold behaviour)
- Defines an entity
- Self-contained, but may depend on other components
  (e.g. view depends on position)

ECS - System

- Holds a collection of components of the same type
- Performs bulk operations
- Data-oriented, rather than object-oriented

ECS - Example Component

```
class MoveableComponent : Component {

    constructor() {
        x = 0.0;
        y = 0.0;
    }
}
```

ECS - Example Entity

```
entity = new Entity();

// now entity knows about its position and that it can move
entity.addComponent(new MoveableComponent());
```

ECS - Example System

```
for (auto moveComponent : allMoveableComponents) {
    moveComponent.x += vx;
    moveComponent.y += vy;
}
```

Entity-Component Model

A popular variation of ECS. The `System` gets merged with `Component`.

Entity-Component Model (Example)

```
class MoveableComponent : Component {

    constructor() {
        x = 0.0;
        y = 0.0;
    }

    update() {
        x += vx;
        y += vy;
    }
}
```

ECS - Examples in Game Engines

- Unreal Engine uses something similar to Entity-Component (EC) model
- FXGL also uses EC model
- Unity is moving from EC to ECS

From now on, for simplicity, when referring to ECS we really mean EC.

ECS - Activity

Design a simple game object (entity) that is built using the EC model. For example, design the component(s) of a ball entity in Pong.

Advantages

- Composition over Inheritance (no inheritance hell)
- Can mix components easily
- Scaling and management (clearly defined dependency)

Disadvantages

- Redundant iteration (e.g. if a component does nothing)
- More verbose API:

```
entity.getComponent(MoveableComponentType).velocity = new V
```

- Need to manage communication between components

ECS - Updating

Every component defines an update:

```
class MoveableComponent : Component {

    constructor(velocity) {
        this.x = 0.0;
        this.y = 0.0;
        this.velocity = velocity;
    }

    update(tpf) {
        x += velocity.x * tpf;
        y += velocity.y * tpf;
    }
}
```

ECS - Example Update

```
entity = new Entity();
entity.addComponent(new MoveableComponent(new Vec2(100, 50)
```

Whenever `update()` is called, `entity` moves.

ECS - Game World

- *Can* be a collection of entities based on its architecture
- Responsible for entity updates and queries

ECS - Game World - Activity

Design a (simple) open world, such as the TESV Skyrim world with its associated game objects. Use the EC model.

Example: `TreeComponent` that will be added to a `tree` entity.

ECS - Game World - Explore

Let’s consider existing EC implementation in FXGL.

Game World - Queries

Example use cases for each?

- By type
- By position
- Random
- By component

Further Reading

These are short (around 5-10mins read) but pack a lot of value!

- ECS back and forth
- Gameprogrammingpatterns Component

Worth watching (but read the above first):

- Overwatch ECS
- Unity ECS

Conclusion

- ECS - a powerful pattern in game dev
- Entities - just generic objects
- Components - add “flavour” to entities
- Systems - update components but may not be present if using just EC
- EC is more manageable for small-medium games

Tutorial

On StudentCentral